Docker & Docker Compose 문제

1. Docker - 종합

다음 설명 중 옳은 것을 `2개` 고르시오.

1. `docker pull` 시 `Digest` 를 사용한다면, 언제나 동일한 이미지를 다운받을 수 있다.
2. `Multi-arch` 이미지는 `Host` 에 저장할 수 있다.
3. `expose` 를 이용하면 포트를 노출시키는 경우, `Host` 머신에서도 컨테이너의 서비스에 접근 할 수 있다.
4. `Network` 는 `1개` 만 사용할 수 있다.
5. `-f` 옵션을 이용하여 사용할 `Dockerfile` 을 지정할 수 있다.

2. Dockerfile

`Dockerfile` 지시어 설명 중 옳지 않은 것은?

1. `RUN` 은 이미지 빌드시에 실행된다.
2. 다른 이미지에서 파일을 복사하기 위해서는 `ADD` 를 사용하여야 한다.
3. 컨테이너에서 사용할 환경변수는 `ENV` 를 이용하여 정의한다.
4. `CMD` 가 여러번 정의된 경우, 가장 마지막에 작성된 것이 사용된다.
5. `build` 시에 사용할 변수는 `ARG` 를 이용하여 정의한다.

3. Dockerfile

다음 `Dockerfile` 과 명령어를 이용하여 이미지를 제작하였을 때, 올바르게 설명한 것을 `2개` 고르시오.

```
# Dockerfile
FROM nginx:stable-bullseye

USER root

RUN apt-get update && apt-get -y upgrade

RUN apt-get install -y curl

RUN mkdir /code

WORKDIR /code

ADD install.sh ./tmp/install_scripts.sh

WORKDIR /root
```

```
EXPOSE 80
```

```
# Build Command
docker build -t quiz project
```

1. 컴퓨터에 따라 사용되는 `nginx:stable-bullseye` 이미지의 `Digest` 값은 달라질 수 있다.
2. `install.sh` 파일은 `/tmp` 아래에 저장된다.
3. 생성된 이미지의 `TAG`는 `project`이다.
4. `install.sh`는 `build` 명령어를 실행한 위치(`Working Directory`)에 있는 파일이 사용된다.
5. 생성된 이미지를 이용하여 컨테이너를 생성하더라도 `-p` 옵션을 사용하지 않는다면 `Host`에서 접근할 수 없다.

4. Dockerfile & Docker Compose

다음 `Dockerfile`을 이용하여 `quiz:4` 이미지를 생성하였다.

```
FROM ubuntu:22.04

CMD ["echo", "3600"]

ENTRYPOINT []
```

다음 `Yaml` 파일을 사용하여 `docker compose up`을 실행하였을 때, 출력되는 값은?

```
# docker-compose.yaml
services:
  server:
    image: quiz:4
    entrypoint: "echo ep"
```

1. `3600`
2. `ep echo 3600`
3. `ep`
4. `3600 ep`
5. 에러가 발생한다.

5. Docker CLI

다음 명령어를 이용하여 컨테이너를 실행하였을 때, 올바르지 않은 것은?

```
$ docker run --name base ubuntu /bin/bash -c "apt-get update && apt-get -y upgrade && apt install -y curl && /bin/bash"
```

1. 이미지는 `ubutu:latest`가 사용된다.
2. 컨테이너 생성 후, 화면에서 로그를 확인할 수 있다.
3. 컨테이너가 실행될 때 컨테이너의 `1번` 프로세스에서 `curl`을 설치한다.
4. `curl`이 설치된 이후에도 컨테이너는 종료되지 않는다.
5. 컨테이너가 종료된 이후에도 `docker logs`를 이용하여 로그를 확인할 수 있다.

6. Docker Compose

다음 설명 중 옳지 않은 것은?

1. `compose version`에 따라서 사용할 수 있는 기능이 달라질 수 있다.
2. 언제나 `docker-compose.yaml` 내 있는 `name`이 프로젝트 이름으로 사용된다.
3. `container_name`이 설정된 경우, 해당 서비스를 구성하는 컨테이너는 언제나 `1개`이다.
4. `pull_policy`가 `build`인 경우, 매번 사용할 이미지를 `build`후 사용한다.
5. `restart`가 `on-failure`인 경우, 정상적으로 종료되지 않은 컨테이너를 매번 재실행한다.

7. Docker Compose CLI

다음 설명 중 옳지 않은 것은?

1. `docker compose up`을 사용하면 현재 디렉토리에 있는 `docker-compose.yaml`이 사용된다.
2. `docker compose restart`를 이용하여, `docker-compose.yaml`에 있는 변경사항을 반영할 수 없다.
3. `docker compose up`을 이용하면 기존에 존재하던 컨테이너도 `매번` 재생성된다.
4. `docker compose logs`를 이용하면 특정 서비스의 로그만 확인할 수도 있다.
5. `docker compose down`를 이용하면 `docker-compose.yaml`에 정의된 서비스만 종료된다.

8. Docker Compose YAML (Anchor & Alias)

다음은 `.env`파일과 `docker-compose.yaml`파일이다.

```
# .env file
FROM=".env.from"
BY=".env.by"
STAGE="local"
```

```
# prod.env file
FROM="Cloudwave"
BY="prod.env"
STAGE="prod"
```

```yaml
# docker-compose.yaml file
version: '3.8'

x-common:
  &common
  restart: always
  volumes:
    - source:/code
  environment:
    &default-env
    BY: "x-common.by"

services:
  ubuntu:
    <<: *common
    image: ubuntu:cloudwave
    build:
      dockerfile_inline: |
        FROM ubuntu:22.04
        ENV FROM=dockerfile
    environment:
      <<: *default-env
      STAGE: "yaml"
      BY: "service.by"
    env_file:
      - prod.env
      - .env
    entrypoint: /bin/bash
    command:
      - -c
      - echo "STAGE = ${STAGE} | FROM = $${FROM} | BY = $${BY}"
    restart: no
    pull_policy: build

volumes:
  source:
```

3개의 파일이 하나의 폴더에 위치하였을 때, `docker compose up`의 출력 값으로 올바른 것은?

1. `STAGE = prod | FROM = dockerfile | BY = prod.env`
2. `STAGE = local | FROM = .env.from | BY = service.by`
3. `STAGE = prod | FROM = .env.from | BY = .env.by`
4. `STAGE = local | FROM = Cloudwave | BY = service.by`
5. `STAGE = local | FROM = dockerfile | BY = service.by`

9. Docker Compose

다음은 `docker-compose.yaml` 이다.

```yaml
# docker-compose.yaml
version: '3.8'
name: 'quiz'

services:
  postgres:
    image: postgres:16.1-bullseye
    networks:
      - db
      - private
    environment:
      - POSTGRES_PASSWORD=mysecretpassword
    volumes:
      - db_data:/var/lib/postgresql/data:rw

  server:
    image: ubuntu:22.04
    networks:
      - db

  pgadmin:
    image: dpage/pgadmin4:7.4
    environment:
      - PGADMIN_DEFAULT_EMAIL=user@sample.com
      - PGADMIN_DEFAULT_PASSWORD=SuperSecret
    networks:
      - db

volumes:
  db_data:
    name: "psql_data"
    external: true

networks:
  private:
    name: "private"
  db:
    name: "db-net"
    external: true
```

위의 `docker-compose.yaml` 을 이용하여 다음 명령어로 프로젝트를 실행하였다. 이때, 옳지 않은 것을 2개 고르세요.

```
$ docker compose -p cloudwave up -d
```

1. 프로젝트의 이름은 `cloudwave` 로 설정된다.
2. `server` 서비스를 구성하는 컨테이너는 생성 후 종료된다.
3. `private` 네트워크는 프로젝트가 실행될 때 생성되지 않는다.

4. 이름이 `psql_data`인 볼륨(`docker volume`)이 생성된다.
5. `postgres`과 `server` 서비스는 서로 네트워크 통신이 가능하다.

10. Docker Compose

아래에 작성된 `configuration`을 이용하여 프로젝트를 실행하려고 한다.

```yaml
name: "quiz"

services:
  backend:
    image: example/database
    volumes:
      - storage:/etc/data
      - config:/mnt/config

  backup:
    image: backup-service
    volumes:
      - ./db-data:/var/lib/backup/data:rw

volumes:
  {NAME}:
    name: "storage"
    external: true
  storage:
    name: "config"
  db-data:
    name: "data"
```

다음 중 `{NAME}`에 들어갈 단어는?

1. `db-data`
2. `config`
3. `quiz_storage`
4. `quiz_config`
5. `external-storage`